Centrum voor Wiskunde en Informatica

REPORT*RAPPORT*

Automata and behaviours in categories of processes

B.P.F. Jacobs

Computer Science/Department of Software Technology

CS-R9607 1996

Report CS-R9607
ISSN 0169-118X

CWI
P.O. Box 94079
1090 GB Amsterdam
The Netherlands

Automata and Behaviours in Categories of Processes

Bart Jacobs

CWI
P.O. Box 94079, 1090 GB Amsterdam, The Netherlands
(bjacobs@cwi.nl)

Abstract

An early result of Goguen [4, 5] describes the fundamental adjunction between categories of deterministic automata and their behaviours. Our first step is to redefine (morphisms in) these categories of automata and behaviours so that a restriction in Goguen's approach can be avoided. Subsequently we give a coalgebraic analysis of this behaviour-realization adjunction; it yields a second generalization to other types of (not only deterministic) automata (and their behaviours).

We further show that our (redefined) categories of automata and behaviours support elementary process combinators like renaming, restriction, parallel composition, replication and feedback (some of which also occur, for example, in the π-calculus). One of the main contributions is that replication $!P$ is defined for an automaton P such that $!P$ is the terminal coalgebra $!P \xrightarrow{\cong} P\|!P$ of the functor $P\|(-)$ "compose with P". The behaviour functor from automata to their behaviours preserves these process combinators, so that the behaviour of a complex automaton can be understood from the behaviour of its components.

AMS Subject Classification (1991): 18C10, 03G30
CR Subject Classification (1991): D.1.3, F.1.1, F.1.2, F.3.2
Keywords & Phrases: automaton, behaviour, realization, process, replication

1. Introduction

Automata have a long and respectable history in computing. In this work we study automata as processes, and focus on process combinators (like composition and replication) that may be defined on automata. In order to describe these combinators properly, we organize automata in suitable categories; this allows us to describe the process combinators as functors acting on this category. In this context we speak of categories of *processes*, precisely because this structure normally found in process calculi exists on these categories. Describing this structure in an abstract categorical framework allows us to describe formally the same structure in other settings (e.g. on categories of non-deterministic automata, or of continuous dynamical systems).

An automaton displays certain behaviour, which is expressed in terms of (alphabet) symbols that are accepted by the automaton. Also such behaviours can be organized in categories. And the definition of an automaton's behaviour yields a "behaviour" functor $\mathcal{B}$ from the category of automata to the category of behaviours. We shall also define process combinators on this category of behaviour, in such a way that this functor $\mathcal{B}$ preserves combinators (i.e. commutes with suitable functors). This yields a form of compositionality: the behaviour of a complex automaton may be understood from the behaviour of its parts.

Joseph Goguen [4, 5] in the early 1970s (and again in [6]) defined categories of (deterministic) automata and behaviours and showed that—under certain restrictions—the behaviour functor $\mathcal{B}$ has a right adjoint $\mathcal{R}$, for realization. This is a fundamental result, bringing a number of "minimal re-

alization" constructions for dynamical systems known at the time under one (categorical) heading. We show that Goguen's restrictions can be avoided by using slightly different categories of automata and behaviours. Our definition takes the contravariance of inputs seriously, and accordingly reverses the morphisms between input alphabets. (Reversing a morphism may seem innocuous, but can be a fundamental (conceptual) step, as for frames and locales, see [10].) Later in Section 3 we shall further generalize the "behaviour-realization" adjunction by extending it to other (than deterministic) automata. The crucial point of this generalization is that automata are coalgebras and that observations are elements of terminal coalgebras (see also [1, 17]). The adjunction ($\mathcal{B} \dashv \mathcal{R}$) then easily follows from terminality.

Automata, described as coalgebras together with an initial state, are used in [8] to give meaning to classes in object-oriented programming, and to explain inheritance via cofree coalgebras. Here we find process combinators on a category of these automata (like renaming, with restriction as special case, various forms of composition, replication !, and also feedback) which are much like in the π-calculus (see e.g. [14]). Actually, the design of the π-calculus was influenced by the object-oriented paradigm, so maybe, in the end, it is not such a surprise that we find π-calculus (or, concurrent object-oriented) structure on these models of classes.

The intuitive meaning of replication $!P$ in the π-calculus is: P infinitely many times in parallel $\|$ with itself. We make this mathematically precise by defining for an automaton P a new automaton $!P$, and by characterizing $!P$ as the terminal coalgebra $!P \xrightarrow{\cong} P\|!P$ of the functor $P\|(-)$ describing composition with P. Then $!P \cong P\|P\|P\|\cdots$. This is in analogy with the characterization of the set $A^{\mathbb{N}} \cong A \times A \times A \times \cdots$ of infinite sequences of A's as the terminal coalgebra of the functor $A \times (-)$ describing the Cartesian product with A. Actually, on deterministic automata we describe two replication functors $\underset{\otimes}{!}$ and $\underset{\oplus}{!}$ associated with two different composition operations $\otimes$ and $\oplus$, see Theorem 2.4.

Sometimes one finds automata organized in 2-categories or in bi-categories, see e.g. [3, Volume A, Chapter VII] or [12, 18], where automata are morphisms. Here we use ordinary categories, with automata as objects. This suits our purposes. The feedback operation that we describe for automata may also be found in [12, 18] (and is standard in dataflow networks and control theory).

It is becoming increasingly clear that coalgebras are fundamental in the study of processes, see e.g. [1, 17, 19] (but also in areas like object-oriented programming [8] or hybrid systems [9]). Coalgebras describe abstract dynamical systems via a state-transition function, and determine the notion bisimulation (observational indistinguishability of states) associated with such systems. Terminal coalgebras are special dynamical systems of "pure" observations in which all bisimilar states are identified. Terminal coalgebras usually form canonical models. The use of coalgebras in this paper in describing automata and their behaviours and in characterizing replication ! further establishes the importance of (terminal) coalgebras in the semantics of processes.

2. Deterministic automata

We shall use some elementary category theory (essentially functors and adjunctions) to organize our results. We shall use the notation $X \times Y$ for the product of two sets (or objects) with first $\pi\colon X \times Y \to X$ and second $\pi'\colon X \times Y \to Y$ projection maps. Dually we use the coproduct (or disjoint union) $X+Y = \{\langle x, 0\rangle \mid x \in X\} \cup \{(\langle y, 1\rangle \mid y \in Y\}$ with first $\kappa\colon X \to X+Y$ and second $\kappa'\colon Y \to X+Y$ coprojections, given by $\kappa(x) = \langle x, 0\rangle$ and $\kappa'(y) = \langle y, 1\rangle$. For a functor $T\colon \mathbf{Sets} \to \mathbf{Sets}$ (or on some other category) a **coalgebra** is a map of the form $\varphi\colon U \to T(U)$. The set U is the "carrier" of the coalgebra, and may be understood as the state space of a dynamical system of which the map φ is the transition function. A morphism of coalgebras from $\varphi\colon U \to T(U)$ to $\psi\colon V \to T(V)$ is a map $f\colon U \to V$

between the state spaces, commuting with the transition functions, i.e. satisfying $\psi \circ f = T(f) \circ \varphi$. A coalgebra is then *terminal* if for an arbitrary coalgebra there is a unique coalgebra map to it.

We first give a concrete description of deterministic automata, which, in the next section, will be subsumed in a more abstract and general framework.

A deterministic automaton consists of a **state space** U together with an **initial state** $u \in U$, an **attribute** $\mathsf{at}\colon U \to B$ and a **procedure** $\mathsf{pr}\colon U \times A \to U$. These two maps form a coalgebra $\langle \mathsf{at}, \mathsf{pr}\rangle\colon U \to B \times U^A$ on U (of the functor $T(X) = B \times X^A$). The set A is usually called the alphabet of the automaton. The set B may be seen as the set of observable outputs. Traditionally in automata theory B is taken to be the two-element set $\{0,1\}$, so that an attribute $U \to \{0,1\}$ may be identified with a subset of U of "final states" (see e.g. [3]; there, automata also have a set of initial states, instead of a single initial state, as we have here). We shall generally refer to an automaton as a structure $\langle u, U \to B \times U^A\rangle$, where we standardly use the notation at and pr for the automaton's attribute and procedure.

(Notice that multiple attributes $U \to B_1$, ..., $U \to B_n$ can be accomodated as a single attribute $U \to (B_1 \times \cdots \times B_n)$ and multiple procedures $U \times A_1 \to U$, ..., $U \times A_m \to U$ as a single procedure $U \times (A_1 + \cdots + A_m) \to U$. Hence there it is no loss of generality in restricting oneself to automata with a single attribute and a single procedure.)

We recall that for given sets A, B the **terminal coalgebra** of the functor $T(X) = B \times X^A$ is the set $B^{A^\star}$ of functions from the set $A^\star$ of finite sequences of A's to B, provided with operations:

$$\begin{array}{lll} \text{attribute} & \mathsf{at}\colon B^{A^\star} \longrightarrow B & h \longmapsto h(\langle\rangle) \\ \text{procedure} & \mathsf{pr}\colon B^{A^\star} \times A \longrightarrow B^{A^\star} & (h,a) \longmapsto \lambda\alpha \in A^\star.\, h(a \cdot \alpha). \end{array}$$

The resulting map $B^{A^\star} \to B \times (B^{A^\star})^A$ is a terminal coalgebra (and thus an isomorphism): for an arbitrary coalgebra $\langle \mathsf{at}, \mathsf{pr}\rangle\colon U \to B \times U^A$ the mediating coalgebra map $U \dashrightarrow B^{A^\star}$ is $x \mapsto \lambda\alpha.\, \mathsf{at}(\overline{\mathsf{pr}}(x,\alpha))$, where $\overline{\mathsf{pr}}\colon U \times A^\star \to U$ is the inductive extension of the procedure $\mathsf{pr}\colon U \times A \to U$. It is defined by $\overline{\mathsf{pr}}(x, \langle\rangle) = x$ and $\overline{\mathsf{pr}}(x, \alpha \cdot a) = \mathsf{pr}(\overline{pr}(x,\alpha), a)$. Then it is easy to see that $\overline{\mathsf{pr}}(x, a \cdot \alpha) = \overline{\mathsf{pr}}(\mathsf{pr}(x,a), \alpha)$. For the special case of $B = \{0,1\}$ (in classical automata theory) the map $U \dashrightarrow \{0,1\}^{A^\star} = \mathcal{P}(A^\star)$ sends an initial state to the subset of $A^\star$ describing the language that is recognized by the automaton.

We define a category **DA** of deterministic automata, and a category **DB** of deterministic behaviours. Notice the (reversed) direction of the arrows between the input alphabets.

2.1. Definition. (i) Let **DA** be the category with

objects deterministic automata $\langle u, U \to B \times U^A\rangle$.

morphisms $\langle u, U \to B \times U^A\rangle \to \langle v, V \to D \times V^C\rangle$ consist of three maps

$$A \xleftarrow{f} C \qquad B \xrightarrow{g} D \qquad U \xrightarrow{\varphi} V$$

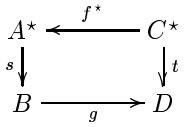
satisfying

$$\begin{array}{ccc} 1 & = & 1 \\ {\scriptstyle u}\downarrow & & \downarrow{\scriptstyle v} \\ U & \xrightarrow[\varphi]{} & V \end{array} \qquad \begin{array}{ccc} U & \xrightarrow{\varphi} & V \\ {\scriptstyle \mathsf{at}}\downarrow & & \downarrow{\scriptstyle \mathsf{at}} \\ B & \xrightarrow[g]{} & D \end{array} \qquad \begin{array}{ccc} U & \xrightarrow{\varphi} & V \\ {\scriptstyle \mathsf{pr}}\downarrow & & \downarrow{\scriptstyle \mathsf{pr}} \\ U^A & \xrightarrow[\varphi^f]{} & V^C \end{array}$$

These diagrams express that $\varphi(u) = v$, $\mathsf{at}(\varphi(x)) = g(\mathsf{at}(x))$ and $\mathsf{pr}(\varphi(x), c) = \varphi(\mathsf{pr}(x, f(c)))$.

(ii) The category **DB** of deterministic behaviours has

objects triples (A, B, s) where A, B are sets and s is a function $s\colon A^\star \to B$.
morphisms $(A, B, s) \to (C, D, t)$ consist of two functions $f\colon C \to A$ and $g\colon B \to D$ satisfying



Identities and composites in these categories are componentwise as in the category **Sets** of sets and functions. Notice the contravariance in the inputs. Goguen [4, 5] uses covariance in the input, and is subsequently forced to put some artifical restrictions on his realization-behaviour adjunction (the "surjectivity on input alphabets", see [5, end of (B)], but also [6, 5.3]). These restriction can be avoided by using the above categories, see Proposition 2.3 below. (Another difference is that Goguen restricts himself to *reachable* automata, for which the map $\overline{\mathsf{pr}}(u, -)\colon A^\star \to U$ is surjective; the adjoint to behaviour $\mathcal{B}$ is then the "Nerode" quotient of the terminal coalgebra $B^{A^\star}$ that we use.)

Renaming and restriction

By the contravariance in the input and covariance in the output (both for automata and for behaviours) we get two forgetful functors

$$\begin{array}{ccc} \mathbf{DA} & \langle u, U \to B \times U^A \rangle \\ \downarrow & \downarrow \\ \mathbf{Sets}^{\mathrm{op}} \times \mathbf{Sets} & (A, B) \end{array} \qquad \text{and} \qquad \begin{array}{ccc} \mathbf{DB} & (A, B, s) \\ \downarrow & \downarrow \\ \mathbf{Sets}^{\mathrm{op}} \times \mathbf{Sets} & (A, B) \end{array}$$

Both these functors are "opfibrations". This means that they come with appropriately universal renaming operations: for a morphism $(f, g)\colon (A, B) \to (C, D)$ in $\mathbf{Sets}^{\mathrm{op}} \times \mathbf{Sets}$ we can rename automata and behaviours over (A, B) to automata and behaviours over (C, D). We write this renaming operation as $(f, g)^*$ in:

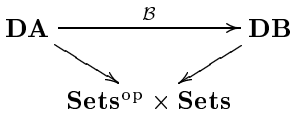
$$\begin{aligned} (f, g)^* \langle u, U \to B \times U^A \rangle &= \langle u, U \to D \times U^C \rangle \quad \text{with operations} \\ &\qquad \mathsf{at}'(x) = g(\mathsf{at}(x)) \text{ and } \mathsf{pr}'(x, c) = \mathsf{pr}(x, f(c)) \\ (f, g)^*(A, B, s) &= (C, D, g \circ s \circ f^\star). \end{aligned}$$

Such renaming is similarly described in terms of fibred categories in [20].

A special case of renaming is *restriction*: if the above maps f and g are the second coprojection $A + C \xleftarrow{\kappa'} C$ and second projection $B \times D \xrightarrow{\pi'} D$, then the functor $(\kappa', \pi')^*$ sends an automaton $\langle u, U \to (B \times D) \times U^{(A+C)} \rangle$ with two attributes $U \to B$, $U \to D$ and two procedures $U \times A \to U$, $U \times C \to U$ to the automaton $\langle u, U \to D \times U^C \rangle$ with one attribute $U \to D$ and one procedure $U \times C \to U$, simply by making the other attribute and procedure invisible. Notice: *restriction is renaming along a projection*, namely along $(A, B) \times (C, D) = (A + C, B \times D) \xrightarrow{(\kappa', \pi')} (C, D)$ in $\mathbf{Sets}^{\mathrm{op}} \times \mathbf{Sets}$.

The behaviour-realization adjunction

2.2. Lemma. *There is a behaviour functor $\mathcal{B}$ in a commuting triangle*



which (strictly) preserves renaming.

Proof. For an automaton $\langle u, U \to B \times U^A \rangle$ we get an associated behaviour function $A^\star \to B$ by $\alpha \mapsto \mathsf{at}(\overline{\mathsf{pr}}(u, \alpha))$. □

Notice that $\mathcal{B}$ is obtained as the unique map to the terminal coalgebra applied to the initial state. This will be generalized in the next section.

2.3. Proposition (After [4, 5]). *The behaviour functor $\mathcal{B}\colon \mathbf{DA} \to \mathbf{DB}$ has a full and faithful right adjoint $\mathcal{R}\colon \mathbf{DB} \to \mathbf{DA}$, called "realization".*

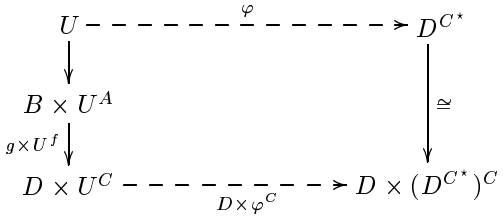
Proof. One defines $\mathcal{R}$ by using terminal coalgebras:

$$\mathcal{R}(C, D, t) = \langle t, D^{C^\star} \xrightarrow{\cong} D \times \left(D^{C^\star}\right)^C \rangle.$$

The adjunction $(\mathcal{B} \dashv \mathcal{R})$ involves a bijective correspondence

$$\frac{\mathcal{B}\langle u, U \to B \times U^A \rangle = (A, B, \mathsf{at}(\overline{\mathsf{pr}}(x, -))) \xrightarrow{(f,g)} (C, D, t)}{\langle u, U \to B \times U^A \rangle \xrightarrow[(f,g,\varphi)]{} \langle t, D^{C^\star} \xrightarrow{\cong} D \times (D^{C^\star})^C \rangle = \mathcal{R}(C, D, t)}$$

In this situation, the map φ is uniquely determined as the mediating coalgebra map in a situation:



Hence the correspondence boils down to:

$$g \circ \mathsf{at}(\overline{\mathsf{pr}}(u, -)) \circ f^\star = t \;\Leftrightarrow\; \varphi(u) = t.$$

And this follows because for $\beta, \gamma \in C^\star$ one proves (by induction on β)

$$\varphi(\overline{\mathsf{pr}}(x, f^\star(\beta)))(\gamma) = \varphi(x)(\beta \cdot \gamma).$$

□

We proceed to define process combinators on the categories **DA** and **DB**. These will be preserved by the behaviour functor $\mathcal{B}\colon \mathbf{DA} \to \mathbf{DB}$, yielding a form of compositionality: the behaviour of a composite automaton may be obtained from the behaviour of its components.

Parallel composition

On the category **DA** of deterministic automata we identify three symmetric monoidal structures $(\otimes, \mathbf{1})$, $(|, \boldsymbol{I})$ and $(\oplus, \mathbf{1})$, each describing a form of parallel composition. There is no communication in these composites, but we do have a separate feedback operation, see below. The different tensors $\otimes, |$ and $\oplus$ describe different ways of putting automata together:

$$\langle u, U \to B \times U^A \rangle \otimes \langle v, V \to D \times V^C \rangle = \langle \langle u, v \rangle, U \times V \to (B \times D) \times (U \times V)^{(A+C)} \rangle$$

with attribute and procedure:

$$\mathsf{at}_\otimes(x, y) = \langle \mathsf{at}(x), \mathsf{at}(y) \rangle \qquad \begin{cases} \mathsf{pr}_\otimes(\langle x, y \rangle, \kappa a) = \langle \mathsf{pr}(x, a), y \rangle \\ \mathsf{pr}_\otimes(\langle x, y \rangle, \kappa' c) = \langle x, \mathsf{pr}(y, c) \rangle. \end{cases}$$

In the $\otimes$-composite the separate automata are put side-by-side: the available attributes and procedures of the two automata are combined: the user can choose which procedure to apply simply by tagging the input by κ or κ'.

The next product is the synchronized product

$$\langle u, U \to B \times U^A \rangle | \langle v, V \to D \times V^C \rangle = \langle \langle u, v \rangle, U \times V \to (B \times D) \times (U \times V)^{(A \times C)} \rangle$$

in which procedures cannot be used separately:

$$\mathsf{at}_|(x, y) = \langle \mathsf{at}(x), \mathsf{at}(y) \rangle \qquad \mathsf{pr}_|(\langle x, y \rangle, \langle a, c \rangle) = \langle \mathsf{pr}(x, a), \mathsf{pr}(y, c) \rangle.$$

Hence in this case, calling the procedure of the $|$-composite means calling both procedures each with their own input. There is a third composite of automata:

$$\langle u, U \to B \times U^A \rangle \oplus \langle v, V \to D \times V^C \rangle = \langle \langle u, v \rangle, U \times V \to (B \times D) \times (U \times V)^{(A \amalg C)} \rangle$$

where $A \amalg C = (A + C) + (A \times C)$. The associated attribute and procedure are:

$$\mathsf{at}_\oplus(x, y) = \langle \mathsf{at}(x), \mathsf{at}(y) \rangle \qquad \begin{cases} \mathsf{pr}_\oplus(\langle x, y \rangle, \kappa(\kappa a)) = \langle \mathsf{pr}(x, a), y \rangle \\ \mathsf{pr}_\oplus(\langle x, y \rangle, \kappa(\kappa' c)) = \langle x, \mathsf{pr}(y, c) \rangle \\ \mathsf{pr}_\oplus(\langle x, y \rangle, \kappa' \langle a, c \rangle) = \langle \mathsf{pr}(x, a), \mathsf{pr}(y, c) \rangle. \end{cases}$$

The $\oplus$-composite is in a sense a combination of $\otimes$ and $|$, since one can choose whether to apply a single procedure, or to apply them both at the same time. But notice that $M \oplus N \neq (M \otimes N) \otimes (M|N)$ for automata $M, N \in \mathbf{DA}$.

The units for these tensors are

$$\mathbf{1} = \langle *, \{*\} \to 1 \times \{*\}^0 \rangle \text{ for } \otimes, \oplus \qquad \text{and} \qquad \boldsymbol{I} = \langle *, \{*\} \to 1 \times \{*\}^1 \rangle \text{ for } |.$$

And since $\mathbf{1} \in \mathbf{DA}$ is the terminal object, one has that $\otimes$ and $\oplus$ are "affine" tensors (with projections, see [7]).

On the category **DB** of deterministic behaviours we can define similar symmetric monoidal structures $(\otimes, \mathbf{1})$, $(|, \boldsymbol{I})$ and $(\oplus, \mathbf{1})$, describing forms of parallel composition for behaviours. We define:

$$\begin{aligned} (A, B, s) \otimes (C, D, t) &= (A + C, B \times D, s \otimes t \colon (A + C)^\star \to (B \times D)) \\ &\qquad \text{with } (s \otimes t)(\sigma) = \langle s(\sigma|_A), t(\sigma|_C) \rangle, \end{aligned}$$

where we write $\sigma|_A \in A^\star$ for the restriction of $\sigma \in (A + C)^\star$ to elements coming from A.

$$\begin{aligned} (A, B, s) | (C, D, t) &= (A + C, B \times D, s | t \colon (A \times C)^\star \to (B \times D)) \\ &\qquad \text{with } (s|t)(\sigma) = \langle s(\pi^\star(\sigma)), t(\pi'^\star(\sigma)) \rangle. \\ (A, B, s) \oplus (C, D, t) &= (A \amalg C, B \times D, s \oplus t \colon (A \amalg C)^\star \to (B \times D)) \\ &\qquad \text{with } (s \oplus t)(\sigma) = \langle s(\sigma|_A), t(\sigma|_C) \rangle. \end{aligned}$$

The units $\mathbf{1} = (0, 1, !)$ and $\boldsymbol{I} = (1, 1, !)$ contain the trivial maps $!$. Again $\mathbf{1} \in \mathbf{DB}$ is terminal. We leave it to the reader to verify that the behaviour functor $\mathcal{B}\colon \mathbf{DA} \to \mathbf{DB}$ (strictly) preserves these symmetric monoidal structures.

Replication

The idea of replication in the π-calculus (see e.g. [14]) is that the replication process $!P$ is the infinite parallel composite $P \| P \| P \| P \| \cdots$, so that $!P = P \| !P$. This suggests that $!P$ is the terminal coalgebra of the operation $P\|(-)$—in analogy with the fact that the set $P^{\mathbb{N}}$ of infinite lists of P's form the terminal coalgebra of $P \times (-)$. In the present setting of deterministic automata and behaviours we shall produce process combinators $\underset{\otimes}{!}$ and $\underset{\oplus}{!}$ forming terminal coalgebras

$$\underset{\otimes}{!}P \xrightarrow{\cong} P \otimes \underset{\otimes}{!}P \qquad \text{and} \qquad \underset{\oplus}{!}P \xrightarrow{\cong} P \oplus \underset{\oplus}{!}P$$

of the functors $P \otimes (-)$ and $P \oplus (-)$, both on $\mathbf{DA}$ and on $\mathbf{DB}$.

We shall concentrate on $\underset{\otimes}{!}$ and leave $\underset{\oplus}{!}$ to the interested reader. We notice that the tensor product $\otimes$ can easily be extended to arbitrary families as

$$\begin{aligned} &\textstyle\bigotimes_{i\in I}\langle u_i, U_i \to B_i \times U_i^{A_i}\rangle \\ &\quad = \quad \langle \lambda i \in I.\, u_i, \textstyle\left(\prod_{i\in I} U_i\right) \to \left(\prod_{i\in I} B_i\right) \times \left(\prod_{i\in I} U_i\right)^{\left(\coprod_{i\in I} A_i\right)}\rangle. \end{aligned}$$

Replication will be defined in terms of this generalized tensor, as product over $\mathbb{N}$ with constant factors:

$$\underset{\otimes}{!}\langle u, U \to B \times U^A\rangle \stackrel{\text{def}}{=} \textstyle\bigotimes_{n\in\mathbb{N}}\langle u, U \to B \times U^A\rangle = \langle \lambda n.\, u, U^{\mathbb{N}} \to B^{\mathbb{N}} \times \left(U^{\mathbb{N}}\right)^{(A\times\mathbb{N})}\rangle.$$

with attribute and procedure

$$\mathsf{at}(h) = \lambda n.\, \mathsf{at}(h(n)) \quad \text{and} \quad \mathsf{pr}(h, \langle a, n\rangle) = \lambda m. \begin{cases} \mathsf{pr}(h(n), a) & \text{if } m = n \\ h(m) & \text{else.} \end{cases}$$

Thus one can think of the replication automaton $\underset{\otimes}{!}\langle u, U \to B \times U^A\rangle$ as $\langle u, U \to B \times U^A\rangle$ infinitely many times in parallel ($\otimes$) with itself. The number n in the procedure input $\langle a, n\rangle$ tells in which component of the infinite composite to apply the original procedure, with input a.

For behaviours we similarly define:

$$\begin{aligned} \underset{\otimes}{!}(A, B, s) \;=\; \textstyle\bigotimes_{n\in\mathbb{N}}(A, B, s) \;=\; &(A \times \mathbb{N}, B^{\mathbb{N}}, \underset{\otimes}{!}(s)\colon (A \times \mathbb{N})^\star \to B^{\mathbb{N}}) \\ &\text{with } \underset{\otimes}{!}(s)(\sigma) = \lambda m.\, s(\sigma|_m), \end{aligned}$$

where $\sigma|_m \in A^\star$ is obtained from $\sigma \in (A \times \mathbb{N})^\star$ by restriction to those a's occurring as $\langle a, m\rangle$ in σ.

2.4. Theorem. *There are functors* $\underset{\otimes}{!} = \bigotimes_{\mathbb{N}}\colon \mathbf{DA} \to \mathbf{DA}$ *and* $\underset{\otimes}{!} = \bigotimes_{\mathbb{N}}\colon \mathbf{DB} \to \mathbf{DB}$*, forming terminal coalgebras*

$$\underset{\otimes}{!}P \xrightarrow{\cong} P \otimes \underset{\otimes}{!}P \qquad \textit{and} \qquad \underset{\otimes}{!}Q \xrightarrow{\cong} Q \otimes \underset{\otimes}{!}Q$$

of the functors $P \otimes (-)\colon \mathbf{DA} \to \mathbf{DA}$ *and of* $Q \otimes (-)\colon \mathbf{DB} \to \mathbf{DB}$*. The behaviour functor* $\mathcal{B}\colon \mathbf{DA} \to \mathbf{DB}$ *strictly preserves such terminal coalgebras.*

The same holds for $\oplus$ *instead of* $\otimes$*.*

Proof. We shall only do the case of $\underset{\otimes}{!}$ on behaviours. We need to show that there is a map $(f, g) : \underset{\otimes}{!}(A, B, s) \xrightarrow{\sim} (A, B, s) \otimes \underset{\otimes}{!}(A, B, s)$ in $\mathbf{DB}$, forming a terminal coalgebra of the functor $(A, B, s) \otimes (-) : \mathbf{DB} \to \mathbf{DB}$. This pair (f, g) will consist of the initial algebra $f\colon A + A \times \mathbb{N} \xrightarrow{\sim} A \times \mathbb{N}$ of the

functor $A + (-)\colon \mathbf{Sets} \to \mathbf{Sets}$, and of the terminal coalgebra $g\colon B^{\mathbb{N}} \xrightarrow{\sim} B \times B^{\mathbb{N}}$ of the functor $B \times (-)\colon \mathbf{Sets} \to \mathbf{Sets}$. These functions f, g are given by:

$$\begin{cases} f(\kappa a) = \langle a, 0\rangle \\ f(\kappa'\langle a, n\rangle) = \langle a, n+1\rangle \end{cases} \qquad \text{and} \qquad g(h) = \langle h(0), \lambda n.\, h(n+1)\rangle.$$

We first have to check that they form a morphism in **DB**:

$$\begin{array}{ccc} \underset{\otimes}{!}(A,B,s) & \xrightarrow{\quad (f,g) \quad} & (A,B,s) \otimes \underset{\otimes}{!}(A,B,s) \\ \| & & \| \\ (A \times \mathbb{N}, B^{\mathbb{N}}, \underset{\otimes}{!}(s)) & & (A + A \times \mathbb{N}, B \times B^{\mathbb{N}}, s \otimes \underset{\otimes}{!}(s)) \end{array}$$

which amounts to showing that for $\sigma \in (A + A \times \mathbb{N})^\star$

$$g(\underset{\otimes}{!}(s)(f^\star(\sigma))) = (s \otimes \underset{\otimes}{!}(s))(\sigma).$$

The right-hand-side yields:

$$\begin{aligned} (s \otimes \underset{\otimes}{!}(s))(\sigma) &= \langle s(\sigma|_A), \underset{\otimes}{!}(s)(\sigma|_{A\times\mathbb{N}})\rangle \\ &= \langle s(\sigma|_A), \lambda m.\, s(\sigma|_{A\times\mathbb{N}}|_m)\rangle. \end{aligned}$$

And the left-hand-side yields the same:

$$\begin{aligned} g(\underset{\otimes}{!}(s)(f^\star(\sigma))) &= \langle \underset{\otimes}{!}(s)(f^\star(\sigma))(0), \lambda m.\, \underset{\otimes}{!}(s)(f^\star(\sigma))(m+1)\rangle \\ &= \langle s(f^\star(\sigma)|_0), \lambda m.\, s(f^\star(\sigma)|_{m+1})\rangle \\ &= \langle s(\sigma|_A), \lambda m.\, s(\sigma|_{A\times\mathbb{N}}|_m)\rangle. \end{aligned}$$

Next assume we have an arbitrary coalgebra $(p, q)\colon (C, D, t) \to (A, B, s) \otimes (C, D, t)$ in **DB**. Then $p\colon A + C \to C$ and $q\colon D \to B \times D$ satisfy $q \circ t \circ p^\star = s \otimes t$, i.e. for $\sigma \in (A + C)^\star$ one has $q(t(q^\star(\sigma))) = \langle s(\sigma|_A), t(\sigma|_C)\rangle$. Since $p\colon A + C \to C$ is an algebra of $A + (-)$ and $q\colon D \to B \times D$ is a coalgebra of $B \times (-)$, we get unique mediating maps $\overline{p}$ and $\overline{q}$ in

$$\begin{array}{ccc} A + C & \xleftarrow{A+\overline{p}} & A + A \times \mathbb{N} \\ {\scriptstyle p}\downarrow & & \cong\downarrow{\scriptstyle f} \\ C & \xleftarrow[\overline{p}]{} & A \times \mathbb{N} \end{array} \qquad \begin{array}{ccc} D & \xrightarrow{\overline{q}} & B^{\mathbb{N}} \\ {\scriptstyle q}\downarrow & & \cong\downarrow{\scriptstyle g} \\ B \times D & \xrightarrow[B\times\overline{q}]{} & B \times B^{\mathbb{N}} \end{array}$$

These mediating maps $\overline{p}, \overline{q}$ are given by iteration:

$$\overline{p}\langle a, n\rangle = p\kappa'^{(n)}(p\kappa a) \qquad \text{and} \qquad \overline{q}(d) = \lambda n.\, \pi q\left(\pi' q^{(n)}(d)\right).$$

We must show that the pair $(\overline{p}, \overline{q})$ is a map of behaviours $(C, D, t) \to \underset{\otimes}{!}(A, B, s)$. This requires for $\sigma \in (A \times \mathbb{N})^\star$ that

$$\overline{q}(t(\overline{p}^\star(\sigma))) = \underset{\otimes}{!}(s)(\sigma) = \lambda m.\, s(\sigma|_m). \tag{$*$}$$

Therefore we need an auxiliary definition and result: for a sequence $\sigma \in (A \times \mathbb{N})^\star$ and a number $m \in \mathbb{N}$ we define a new sequence $\sigma - m \in (A \times \mathbb{N})^\star$ by subtracting m from n in elements $\langle a, n\rangle$ in σ, if $n \geq m$, and by removing $\langle a, n\rangle$ otherwise. The formal definition is by induction:

$$\langle\rangle - m = \langle\rangle \qquad \text{and} \qquad (\sigma \cdot \langle a, n\rangle) - m = \begin{cases} (\sigma - m) \cdot \langle a, n - m\rangle & \text{if } n \geq m \\ \sigma - m & \text{else.} \end{cases}$$

Claim. For each $\sigma \in (A \times \mathbb{N})^\star$ and $m \in \mathbb{N}$ one has $(\pi' q)^{(m)}(t(\overline{p}^\star(\sigma))) = t(\overline{p}^\star(\sigma - m))$.

The proof is by induction on m. The case $m = 0$ is easy, since $\sigma - 0 = \sigma$. And:

$$\begin{aligned}
\Big((\pi' q)^{(m+1)} \circ t \circ \overline{p}^\star\Big)(\sigma) &\overset{\text{(IH)}}{=} \Big(\pi' \circ q \circ t \circ \overline{p}^\star\Big)(\sigma - m) \\
&= \Big(\pi' \circ q \circ t \circ p^\star \circ (A + \overline{p})^\star \circ f^{-1\star}\Big)(\sigma - m) \\
&= t\Big(((A + \overline{p})^\star f^{-1\star}(\sigma - m))|_C\Big) \\
&= t\Big(\overline{p}^\star(f^{-1\star}(\sigma - m)|_{A \times \mathbb{N}})\Big) \\
&= t\Big(\overline{p}^\star((\sigma - m) - 1)\Big) \\
&= t\Big(\overline{p}^\star(\sigma - (m+1))\Big).
\end{aligned}$$

We are now in a position to prove $(*)$:

$$\begin{aligned}
\Big(\overline{q} \circ t \circ \overline{p}^\star\Big)(\sigma)(m) &= \Big(\pi \circ q \circ (\pi' q)^{(m)} \circ t \circ \overline{p}^\star\Big)(\sigma) \\
&= \Big(\pi \circ q \circ t \circ \overline{p}^\star\Big)(\sigma - m) \quad \text{by the above claim} \\
&= \Big(\pi \circ q \circ t \circ p^\star \circ (A + \overline{p})^\star \circ f^{-1\star}\Big)(\sigma - m) \\
&= s\Big(((A + \overline{p})^\star f^{-1\star}(\sigma - m))|_A\Big) \\
&= s\Big((f^{-1\star}(\sigma - m))|_A\Big) \\
&= s\Big((\sigma - m)|_0\Big) \\
&= s\Big(\sigma|_m\Big).
\end{aligned}$$

□

We conclude this part with three remarks. (1) The operations $!_{\otimes}$ and $!_{\oplus}$ do not extend to comonads, since there are no diagonals. This means that they are π-calculus !'s and not linear logic !'s. (2) In defining a replication functor $!_{\oplus}$ associated with the parallel composition $\oplus$ one uses the initial algebra $A \times (1 + A)^\star$ of the functor $A \amalg (-)\colon \mathbf{Sets} \to \mathbf{Sets}$. (3) Since morphisms in the categories **DA** and **DB** are contravariant in the first (input) coordinate, the terminal coalgebras given by $!_{\otimes}$ and $!_{\oplus}$ are obtained by taking initial algebras in the input, and terminal coalgebras in the output. This works for $\otimes$ and $\oplus$, but not for $|$ since the initial algebra of $A \times (-)$ is the trivial (empty) set.

Feedback

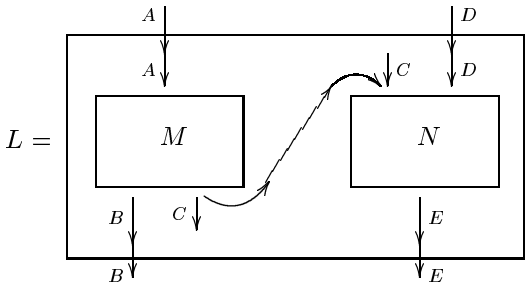
We shall describe for a fixed set A, serving both as input and output, an operation F_A for "feedback at A". It applies to automata (and behaviours) with A both as set of inputs and as set of outputs. The result of applying the feedback operation F_A is an automaton (or behaviour) with the same input and output sets, but with different operation: if a procedure is called, it is executed whereupon the resulting attribute value at A is fed back into the system by calling the procedure with A as input. Formally, F_A sends

$$\langle u, U \xrightarrow{\langle \mathsf{at}, \mathsf{pr} \rangle} (A \times B) \times U^{(A+C)} \rangle \longmapsto \langle u, U \xrightarrow{\langle \mathsf{at}, \mathsf{pr}' \rangle} (A \times B) \times U^{(A+C)} \rangle$$

with new procedure pr' given on $x \in U$ and $d \in A + C$ by

$$\mathsf{pr}'(x, d) = \mathsf{pr}\big(\mathsf{pr}(x, d), \kappa\pi\mathsf{at}(\mathsf{pr}(x, d))\big).$$

2.5. Example. For automata $M = \langle u, U \to (B \times C) \times U^A \rangle$ and $N = \langle v, V \to E \times V^{(C+D)} \rangle$ we wish to describe the composite automaton L given by



as process term (involving M and N). This is done as follows.

1. First we form the composite $M \otimes N$. This means that we "physically" put M and N side-by-side. The result is an automaton $\langle \langle u, v \rangle, U \times V \to ((B \times C) \times E) \times (U \times V)^{(A+(C+D))} \rangle$ with input set $A + (C + D)$ and output set $(B \times C) \times E$.

2. Next, in order to apply the feedback functor F_C we have to rearrange the input and output sets so that C is in first position. This is done by renaming along the rearrangement isomorphisms

$$C + (A + D) \xrightarrow[\cong]{\varphi} A + (C + D) \qquad (B \times C) \times E \xrightarrow[\cong]{\psi} C \times (B \times E)$$

   We thus get an automaton $(\varphi, \psi)^*(M \otimes N)$ with input set $C+(A+D)$ and output set $C \times (B \times E)$.

3. Now we can apply feedback F_C at C. This yields a new automaton $F_C(\varphi, \psi)^*(M \otimes N)$, still with input set $C + (A + D)$ and output set $C \times (B \times E)$.

4. Finally, we make the C input and ouput invisible from the outside by restriction, i.e. by renaming along the coprojection $\kappa' \colon (A+D) \to C+(A+D)$ and the projection $\pi' \colon C \times (B \times E) \to (B \times E)$. Thus we get a process term

$$L = (\kappa', \pi')^* F_C (\varphi, \psi)^* (M \otimes N)$$

   describing the automaton L in the above diagram, with $A + D$ as input and $B \times E$ as output. It is easy to see that applying the procedure pr_L of L with input $a \in A$ (formally with $\kappa a \in A + D$) involves both procedures of M and N in

$$\mathsf{pr}_L(\langle x, y \rangle, \kappa a) = \langle \mathsf{pr}_M(x, a), \mathsf{pr}_N(y, \pi' \mathsf{at}(\mathsf{pr}_M(x, a))) \rangle.$$

   This shows how the C-output of M is fed back into the C-input of N through feedback.

This feedback operation F_A at A can be made functorial: form the category $\mathbf{DA}(A)$ by pullback:

$$\begin{array}{ccc} \mathbf{DA}(A) & \longrightarrow & \mathbf{DA} \\ \downarrow & \lrcorner & \downarrow \\ \mathbf{Sets}^{\mathrm{op}} \times \mathbf{Sets} & \xrightarrow{(X,Y) \mapsto (A+X, A \times Y)} & \mathbf{Sets}^{\mathrm{op}} \times \mathbf{Sets} \end{array}$$

This means that the category $\mathbf{DA}(A)$ has as objects automata of the form $\langle u, U \to (A \times B) \times U^{(A+C)} \rangle$. Then one can extend F_A to a functor $F_A \colon \mathbf{DA}(A) \to \mathbf{DA}(A)$. And a similar feedback functor $F_A \colon \mathbf{DB}(A) \to \mathbf{DB}(A)$ can be defined on behaviours; and commutation with behaviour $\mathcal{B}$ can be established.

3. Other types of automata

So far we have restricted ourselves to deterministic automata. In this section we consider automata of the form

$$\langle u, U \to B \times T(U)^A \rangle$$

where T is a functor **Sets** $\to$ **Sets**, serving as parameter. Typical examples are:

$T = id$	in this manner we recover deterministic automata
$T = 1 + (-)$	to get "partial" automata
$T = \mathcal{P}_f(-)$	for (finitely) non-deterministic automata
$T = (-)^\star$	for "ordered" non-deterministic automata.
$\vdots$	

Notice that all this examples of T are "computational monads" in the sense of [15] (which come equipped with a "strength" map $X \times T(Y) \to T(X \times Y)$), describing some type of computation.

For an arbitrary functor $T\colon$ **Sets** $\to$ **Sets** we can form a category **Aut**(T) of "T-automata" with

objects automata $\langle u, U \to B \times T(U)^A \rangle$.

morphisms $\langle u, U \to B \times T(U)^A \rangle \to \langle v, V \to D \times T(V)^C \rangle$ consist of triples of maps $f\colon C \to A$, $g\colon B \to D$ and $\varphi\colon U \to V$ such that $\varphi(u) = v$ and φ a map of coalgebras in:

$$\begin{array}{ccc} U & \xrightarrow{\varphi} & V \\ \downarrow & & \downarrow \\ B \times T(U)^A & & \\ {\scriptstyle g \times T(U)^f}\downarrow & & \\ D \times T(U)^C & \xrightarrow[D \times T(\varphi)^C]{} & D \times T(V)^C \end{array}$$

In the remainder of this section we shall assume that this functor T is such that for each pair of sets A, B we have a terminal coalgebra

$$P(A,B) \xrightarrow{\cong} B \times T(P(A,B))^A \qquad \text{of the functor} \qquad X \mapsto B \times T(X)^A.$$

For the examples of T listed above, such terminal coalgebras do indeed exist.

Under this assumption we can define a category **Beh**(T) of "T-behaviours". It has

objects triples (A, B, s), where $s \in P(A, B)$.

morphisms $(A, B, s) \to (C, D, t)$ consist of pairs of functions $f\colon C \to A$ and $g\colon B \to D$ satisfying $P(f, g)(s) = t$, where $P(f, g)\colon P(A, B) \to P(C, D)$ is the unique mediating map in the situation:

$$\begin{array}{ccc} P(A,B) & \overset{P(f,g)}{\dashrightarrow} & P(C,D) \\ {\scriptstyle\cong}\downarrow & & \downarrow \\ B \times T(P(A,B))^A & & \downarrow{\scriptstyle\cong} \\ {\scriptstyle g \times T(P(A,B))^f}\downarrow & & \downarrow \\ D \times T(P(A,B))^C & \underset{D \times T(P(f,g))^C}{\dashrightarrow} & D \times T(P(C,D))^C \end{array}$$

(Both $\mathbf{Aut}(T)$ and $\mathbf{Beh}(T)$ are opfibred over $\mathbf{Sets}^{\mathrm{op}} \times \mathbf{Sets}$, so we have renaming and restriction functors.)

There is then an obvious (full and faithful) realization functor $\mathcal{R}\colon \mathbf{Beh}(T) \to \mathbf{Aut}(T)$ using terminal coalgebras:

$$\begin{cases} (A,B,s) & \mapsto & \langle s, P(A,B) \xrightarrow{\cong} B \times T(P(A,B))^A \rangle \\ (f,g) & \mapsto & (f,g,P(f,g)) \end{cases}$$

And in the reverse direction we have a behaviour functor $\mathcal{B}\colon \mathbf{Aut}(T) \to \mathbf{Beh}(T)$ with

$$\begin{cases} \langle u, U \to B \times T(U)^A \rangle & \mapsto & (A,B,\overline{U}(u)) \\ (f,g,\varphi) & \mapsto & (f,g) \end{cases}$$

where $\overline{U}$ is the unique map to the terminal coalgebra:

$$\begin{array}{ccc} U & \overset{\overline{U}}{\dashrightarrow} & P(A,B) \\ \downarrow & & \downarrow \cong \\ B \times T(U)^A & \underset{B \times T(\overline{U})^A}{\dashrightarrow} & B \times T(P(A,B))^A \end{array}$$

3.1. Proposition. *The behaviour functor $\mathcal{B}\colon \mathbf{Aut}(T) \to \mathbf{Beh}(T)$ is left adjoint to the (full and faithful) realization functor $\mathcal{R}\colon \mathbf{Beh}(T) \to \mathbf{Aut}(T)$.*

Notice that the earlier Proposition 2.3 is a special case where T is the identity functor. We now present a more abstract proof using terminality.

Proof. For $\langle u, U \to B \times T(U)^A \rangle \in \mathbf{Aut}(T)$ and $(C,D,t) \in \mathbf{Beh}(T)$ we have a bijective correspondence

$$\frac{(A,B,\overline{U}(u)) \xrightarrow{(f,g)} (C,D,t) \quad \text{in } \mathbf{Beh}(T)}{\langle u, U \to B \times T(U)^A \rangle \xrightarrow[(f,g,\varphi)]{} \langle t, P(C,D) \xrightarrow{\cong} D \times T(P(C,D))^C \rangle \quad \text{in } \mathbf{Aut}(T)}$$

since in this situation the map $\varphi\colon U \to P(C,D)$ is determined as $P(f,g) \circ \overline{U}$ in:

$$\begin{array}{ccccc} U & & \xrightarrow{\quad\varphi\quad} & & P(C,D) \\ & \searrow^{\overline{U}} & & \nearrow_{P(f,g)} & \\ \downarrow & & P(A,B) & & \downarrow \cong \\ B \times T(U)^A & \longrightarrow & B \times T(P(A,B))^A & & \\ {\scriptstyle g \times T(U)^f}\downarrow & & \downarrow{\scriptstyle g \times T(P(A,B))^f} & & \\ D \times T(U)^C & \longrightarrow & D \times T(P(A,B))^C & \longrightarrow & D \times T(P(C,D))^C \end{array}$$

Hence $P(f,g)(\overline{U}(u)) = t \Leftrightarrow \varphi(u) = t$, as required. □

A next step is to consider process combinators on these categories $\mathbf{Aut}(T)$ and $\mathbf{Beh}(T)$ parametrized by the functor T describing a notion of computation. The tensor product $\otimes$ can be defined if T is strong, and $|, \oplus$ require that T is a commutative monad. Instead of describing these matters in full generality, we sketch for the special case of $T = (-)^\star$ parallel composition $\otimes$ and the associated replication operation $\overset{!}{\underset{\otimes}{}}$ for ordered non-deterministic automata. (They can also be defined for their behaviours.)

$$\begin{aligned}&\langle u, U \to B \times (U^\star)^A\rangle \otimes \langle v, V \to D \times (V^\star)^C\rangle\\ &= \quad \langle\langle u, v\rangle, U \times V \to (B \times D) \times ((U \times V)^\star)^{(A+C)}\rangle\end{aligned}$$

with attribute and procedure:

$$\mathsf{at}(x,y) = \langle \mathsf{at}(x), \mathsf{at}(y)\rangle \qquad \left\{\begin{array}{l} \mathsf{pr}(\langle x, y\rangle, \kappa a) = \langle (x_0, y), \ldots, (x_m, y)\rangle \\ \qquad \text{if } \mathsf{pr}(x, a) = \langle x_0, \ldots, x_m\rangle \\ \mathsf{pr}(\langle x, y\rangle, \kappa' c) = \langle (x, y_0), \ldots, (x, y_m)\rangle \\ \qquad \text{if } \mathsf{pr}(y, c) = \langle y_0, \ldots, y_m\rangle. \end{array}\right.$$

(This procedure definition makes use of strength.) And:

$$\underset{\otimes}{!}\langle u, U \to B \times (U^\star)^A\rangle \quad = \quad \langle \lambda n.\, u, U^{\mathbb{N}} \to B^{\mathbb{N}} \times \left((U^{\mathbb{N}})^\star\right)^{(A \times \mathbb{N})}\rangle$$

with attribute and procedure

$$\begin{aligned} \mathsf{at}(h) &= \lambda n.\, \mathsf{at}(h(n)) \\ \mathsf{pr}(h, \langle a, n\rangle) &= \langle h[x_0/n], \ldots, h[x_m/n]\rangle \\ &\qquad \text{if } \mathsf{pr}(h(n), a) = \langle x_0, \ldots, x_m\rangle \\ &\qquad \text{where } h[x_i/n] = \lambda m. \left\{\begin{array}{ll} x_i & \text{if } m = n \\ h(m) & \text{else.}\end{array}\right.\end{aligned}$$

We conclude by simply stating (without giving further details) that a result like Theorem 2.4 holds for these ordered non-deterministic automata. This establishes the characterization of replication ! via terminal coalgebras in another (slightly different) context.

4. Concluding remarks and future work

Via a new definition of categories of automata and of behaviours we have polished and generalized Goguen's fundamental behaviour-realization adjunction ($\mathcal{B} \dashv \mathcal{R}$). Further, we have identified certain process structure in these categories (which is preserved by the (compositional) behaviour functor $\mathcal{B}$).

Automata theory may be seen (following Arbib) as part of the broader area of system theory. Future work is directed at a similar (coalgebraic) analysis of other "dynamical" systems—in the broader sense, involving general state-base computation—such as (linear) dynamical systems [11, 16, 13] or hybrid systems [2] involving continuous behaviour. The abstractness of our approach opens the possibility to identify the operations of (modern) process theory in the (older) field of dynamical systems; and this can serve as a basis for a suitable (modular) language describing complex systems (see the simple illustration in Example 2.5).

References

1. P. Aczel. *Non-well-founded sets.* CSLI Lecture Notes 14, Stanford, 1988.
2. R. Alur, C. Courcoubetis, N. Halbwachs, T.A. Henzinger, P.-H. Ho, X. Nicollin, A. Olivero, J. Sifakis, and S. Yovine. The algorithmic analysis of hybrid systems. *Theor. Comp. Sci.*, 138(1):3–34, 1995.
3. S. Eilenberg. *Automata, Languages and Machines.* Academic Press, 1974. 2 volumes.
4. J.A. Goguen. Minimal realization of machines in closed categories. *Bull. Amer. Math. Soc.*, 78(5):777–783, 1972.
5. J.A. Goguen. Realization is universal. *Math. Syst. Theor.*, 6(4):359–374, 1973.
6. J.A. Goguen. A categorical manifesto. *Math. Struct. Comp. Sci.*, 1(1):49–67, 1991.
7. B. Jacobs. Semantics of weakening and contraction. *Ann. Pure & Appl. Logic*, 69(1):73–106, 1994.

8. B. Jacobs. Inheritance and cofree constructions. CWI Techn. Rep. CS-R9564, available from `ftp.cwi.nl` in `/pub/bjacobs`. To appear in *European Conference on object-oriented programming* (ECOOP 1996), Springer LNCS, 1996.
9. B. Jacobs. Coalgebraic specifications and models of deterministic hybrid systems. In M. Wirsing, editor, *Algebraic Methods and Software Technology*, Lect. Notes Comp. Sci. Springer, Berlin, 1996, to appear.
10. P.T. Johnstone. *Stone Spaces.* Number 3 in Studies in Adv. Math. Cambridge Univ. Press, 1982.
11. R.E. Kalman, P.L. Falb, and M.A. Arbib. *Topics in Mathematical System Theory.* McGraw-Hill Int. Series in Pure & Appl. Math., 1969.
12. P. Katis, N. Sabadini, and R.F.C. Walters. The bicategory of circuits. *Journ. Pure Appl. Algebra*, 1996. To appear.
13. D.G. Luenberger. *Introduction to Dynamic Systems. Theory, Models and Applications.* John Wiley & Sons, 1979.
14. R. Milner. Functions as processes. *Math. Struct. Comp. Sci.*, 2(2):119–141, 1993.
15. E. Moggi. Notions of computation and monads. *Inf. & Comp.*, 93(1):55–92, 1991.
16. L. Padulo and M.A. Arbib. *System Theory. A unified state-space approach to continuous and discrete time systems.* Saunders, 1974.
17. J. Rutten and D. Turi. On the foundations of final semantics: non-standard sets, metric spaces and partial orders. In J.W. de Bakker, W.P. de Roever, and G. Rozenberg, editors, *Semantics: Foundations and Applications*, number 666 in Lect. Notes Comp. Sci., pages 477–530. Springer, Berlin, 1993.
18. E.W. Stark. Compositional relational semantics for indeterminate dataflow networks. In D.H. Pitt, A. Poigné, and D.E. Rydeheard, editors, *Category Theory and Computer Science*, number 389 in Lect. Notes Comp. Sci., pages 52–74. Springer, Berlin, 1989.
19. D. Turi. *Functorial operational semantics and its denotational dual.* PhD thesis, Free Univ. Amsterdam, 1996.
20. G. Winskel and M. Nielsen. Models for concurrency. In S. Abramski, Dov M. Gabbai, and T.S.E. Maibaum, editors, *Handbook of Logic in Computer Science*, volume ?? Oxford Univ. Press, 199?. To appear.